Probability Data Structure Template Library

Hamed Zaghaghi

Jun 18, 2020

CONTENTS:

1 Membership **1**
1.1 Bloom Filter 1
1.2 Counting Bloom Filter 2
1.3 Quotient Filter 3
1.4 Cuckoo Filter 4
1.5 Supported Methods: 5

Index **7**

CHAPTER

ONE

MEMBERSHIP

1.1 Bloom Filter

template<std::size_t **HC**, std::size_t **MC**, template<typename...> class **HF** = mmh3_hash_factory, typename **T** = std::string, typename **S** =
class `pdstl::bloom_filter` : **public** pdstl::membership<*T*>

Standard Bloom Filter.

bloom_filter class implements bloom filter algorithm for solving membership problem.

Template Parameters

- `HC`: - Number of hash functions
- `MC`: - Number of memory bits
- `HF`: - Hash factory method class (default: pdstl::mmh3_hash_factory)
- `T`: - Element type which will be inserted into bloom filter (default: std::string)
- `S`: - Hash output size (default: uint32_t)

Subclassed by *pdstl::counting_bloom_filter< HC, MC, C, HF, T, S >*

Public Functions

bloom_filter()

Default constructor.

void **insert**(**const** *T &item*) **override**

insert an item into bloom filter

Insert item into bloom filter.

Parameters

- `item`: - the item to insert into the bloom filter.

void **erase**(**const** *T &item*) **override**

erase an item from bloom filter

Erase is not supported in standard bloom filter. Calling this method will throw an exception

Parameters

- `item`: - the item to erase from filter.

void **clear**() **override**
clear filter and resets its internal memory.

bool **contains**(**const** *T &item*) **const** **override**
Check the item and report that it's in the filter or not.

Return false if the item is not in the filter, true if item may be in the filter.

Parameters

- item: - the item to check for existence.

1.2 Counting Bloom Filter

template<std::size_t **HC**, std::size_t **MC**, typename **C** = uint16_t, template<typename...> class **HF** = mmh3_hash_factory, typename **T** = s
class pdstl::**counting_bloom_filter** : **public** pdstl::*bloom_filter*<*HC*, *MC*, *HF*, *T*, *S*>
Counting Bloom Filter.

counting_bloom_filter class implements counting filter algorithm for solving membership problem.

Template Parameters

- HC: - Number of hash functions
- MC: - Number of memory bits
- C: - Type of counter (default: uint16_t)
- HF: - Hash factory method class (default: pdstl::mmh3_hash_factory)
- T: - Element type which will be inserted into counting bloom filter (default: std::string)
- S: - Hash output size (default: uint32_t)

Public Functions

counting_bloom_filter()
Default constructor.

void **insert**(**const** *T &item*) **override**
Insert item into counting bloom filter.

Parameters

- item: - the item to insert into the bloom filter.

void **erase**(**const** *T &item*) **override**
Erase item from counting bloom filter.

Parameters

- item: - the item to erase from filter.

void **clear**() **override**
Clear filter and resets its internal memory.

1.3 Quotient Filter

template<std::size_t **F**, std::size_t **Q**, template<typename...> class **HF** = mmh3_hash_factory, typename **T** = std::string, typename **S** = ui
class `pdstl::quotient_filter` : **public** pdstl::membership<*T*>

Quotient Filter.

quotient_filter class implements quotient filter algorithm for solving membership problem.

Template Parameters

- `F`: - Fingerprint bits, must be smaller than or equal to hash output size
- `Q`: - Number of bits for quotient part, remainder bit size is (F - Q)
- `HF`: - Hash factory method class (default: pdstl::mmh3_hash_factory)
- `T`: - Element type which will be inserted into quotient filter (default: std::string)
- `S`: - Hash output type (default: uint32_t)

Public Functions

quotient_filter()

Default constructor.

void **insert**(**const** *T &item*) **override**

insert an item into quotient filter

Parameters

- `item`: - the item to insert into the quotient filter.

void **erase**(**const** *T &item*) **override**

Erase is not supported in standard quotient filter. will throw an exception.

Parameters

- `item`: - the item to erase from filter.

void **clear**() **override**

Clear filter and resets its internal memory.

bool **contains**(**const** *T &item*) **const override**

Check the item and report that it's in the filter or not.

Return false if the item is not in the filter, true if item may be in the filter.

Parameters

- `item`: - the item to check for existence.

1.4 Cuckoo Filter

template<typename **CT**, template<typename...> class **HF** = mmh3_hash_factory, typename **T** = std::string, typename **S** = uint32_t>
class `pdstl::cuckoo_filter` : **public** pdstl::membership<*T*>

Cuckoo Filter.

cuckoo_filter class implements cuckoo filter algorithm for solving membership problem.

Template Parameters

- `CT`: - cuckoo table
- `HF`: - Hash factory method class (default: pdstl::mmh3_hash_factory)
- `T`: - Element type which will be inserted into cuckoo filter (default: std::string)
- `S`: - Hash output type (default: uint32_t)

Public Functions

cuckoo_filter(size_t *num_buckets*, size_t *max_kicks*)

Default constructor.

Parameters

- `num_buckets`: - number of buckets in this filter

void **insert**(**const** *T &item*) **override**

insert an item into quotient filter

Parameters

- `item`: - the item to insert into the quotient filter.

void **erase**(**const** *T &item*) **override**

Erase is not supported in standard quotient filter. will throw an exception.

Parameters

- `item`: - the item to erase from filter.

void **clear**() **override**

Clear filter and resets its internal memory.

bool **contains**(**const** *T &item*) **const override**

Check the item and report that it's in the filter or not.

Return false if the item is not in the filter, true if item may be in the filter.

Parameters

- `item`: - the item to check for existence.

1.5 Supported Methods:

Filter Name	Insert	Delete
Bloom Filter	Supported	Not Supported
Counting Bloom Filter	Supported	Supported
Quotient Filter	Supported	Not Implemented
Cuckoo Filter	Supported	Supported

INDEX

P

`pdstl::bloom_filter` (*C++ class*), 1
`pdstl::bloom_filter::bloom_filter` (*C++ function*), 1
`pdstl::bloom_filter::clear` (*C++ function*), 1
`pdstl::bloom_filter::contains` (*C++ function*), 2
`pdstl::bloom_filter::erase` (*C++ function*), 1
`pdstl::bloom_filter::insert` (*C++ function*), 1
`pdstl::counting_bloom_filter` (*C++ class*), 2
`pdstl::counting_bloom_filter::clear` (*C++ function*), 2
`pdstl::counting_bloom_filter::counting_bloom_filter` (*C++ function*), 2
`pdstl::counting_bloom_filter::erase` (*C++ function*), 2
`pdstl::counting_bloom_filter::insert` (*C++ function*), 2
`pdstl::cuckoo_filter` (*C++ class*), 4
`pdstl::cuckoo_filter::clear` (*C++ function*), 4
`pdstl::cuckoo_filter::contains` (*C++ function*), 4
`pdstl::cuckoo_filter::cuckoo_filter` (*C++ function*), 4
`pdstl::cuckoo_filter::erase` (*C++ function*), 4
`pdstl::cuckoo_filter::insert` (*C++ function*), 4
`pdstl::quotient_filter` (*C++ class*), 3
`pdstl::quotient_filter::clear` (*C++ function*), 3
`pdstl::quotient_filter::contains` (*C++ function*), 3
`pdstl::quotient_filter::erase` (*C++ function*), 3
`pdstl::quotient_filter::insert` (*C++ function*), 3
`pdstl::quotient_filter::quotient_filter` (*C++ function*), 3